CS 111 - Homework 2

Deadline

Problem 7 - correctly answering the CS 111 Syllabus Confirmation and Reading Questions that are posted on the course Moodle site -- is due by **11:59 pm** on **Sunday, September 4, 2016**

The final versions of **Problems 1-6** are due by **11:59 pm** on **Friday, September 9, 2016**

How to submit

You complete Problem 7 - answering the CS 111 Syllabus Confirmation and Reading Questions - on the course Moodle site.

Each time you would like to submit your work for Problems 1-6:

- save your current Definitions window contents in a file with the name `111hw2.rkt`
  - Note: please use that *exact* name -- do not change the case, add blanks, etc. If I search for a file with that name in your submission, I want it to show up! 8-)
- transfer/copy that file to a directory on `nrs-labs.humboldt.edu` (preferably in a folder/directory named `111hw2`)
  - If you are in a campus lab, you can do so by copying them to a folder on the `U:` drive
  - If you are not in a campus lab, you can do so by using `sftp` (secure file transfer) and connecting to `nrs-labs.humboldt.edu`, and then transferring them.

    Graphical versions of `sftp` include WinSCP and Secure Shell Transfer Client for Windows, Cyberduck and Fugu for Mac OS X, and FileZilla that has versions for both Windows and Mac OS X. (Mac OS X and Linux also come with a command-line `sftp` -- an intro to this is included in the posted handout, "useful details - `sftp`, `ssh`, and `~st10/111submit`", available on the public course web site in the References section.)
  - Whichever version of `sftp` you are using, use a **host name** of `nrs-labs.humboldt.edu` if you are not in a campus lab (you can get away with just `nrs-labs` for the host name in a campus lab), your campus username, and when you are prompted for it, your campus password. IF you need to give a port number, give a port number of **22**.
- Now that your file is on nrs-labs.humboldt.edu, you need to log onto nrs-labs.humboldt.edu using `ssh`, so you can submit your file to me.
  - In BSS 313, the Windows computers have a graphical version of `ssh` called PuTTY. There is a graphical version of `ssh` for Mac OS X named Jelly FiSSH, (although Mac OS X and Linux also come with command-line `ssh`, and an intro to this is also included in the "useful details - sftp, ssh, and ~st10/111submit" handout.)
  - Again, whichever version of `ssh` you are using, use a **host name** of `nrs-labs.humboldt.edu` if you are not in a campus lab (you can get away with just `nrs-labs` for the host name in a campus lab), your campus username, and when you are prompted for it, your campus password. IF you need to give a port number, give a port number of **22**.
- WHILE you are logged onto nrs-labs:
  - IF you saved your file in a folder, use `cd` to change to the folder/directory where you saved it -- for example, if you saved it in the folder `111hw2`, then you would go to that directory by saying:

```
cd 111hw2
```

  - use the `ls` command to make sure your `111hw2.rkt` file is really there:

    ```
    ls
    ```

  - type the command:

    ```
    ~st10/111submit
    ```

    ...and when asked, enter a homework number of `2`

    ...and when asked, enter `y`, you do want to submit all files with an appropriate suffix (I don't mind getting some extra files, as long as I also get `111hw2.rkt`; I'd rather receive too many files than too few due to typos.)

    ...make sure to carefully check the list of files submitted, and make SURE it lists `111hw2.rkt` as having been submitted! (The most common error is to try to run `~st10/111submit` while in a different directory than where your files are...)

- (we practiced the above during the Week 1 Lab, on Friday, August 26 -- ASK ME if this is still not clear, or if you have any problems with submission! If you get stuck, E-MAIL me your `111hw2.rkt` file as an e-mail attachment before the deadline, and then submit it as soon after that as you are able.)
  - I am happy to walk through submitting files with you -- just come by my office during office hours, or e-mail me to set up an appointment.

Purpose

To make sure you have read over the course syllabus, and to practice using the design recipe to write and test functions.

Important notes

- Signature and purpose statement comments are ONLY required for **function definitions** -- you do NOT write them for named constants or for non-function-definition compound expressions.
- Remember: A **signature** in Racket is written in a comment, and it includes the name of the function, the **types** of expressions it expects, and the **type** of expression it returns. This should be written as discussed in class (and you can find examples in the posted in-class examples). For example,

  ```
  ; signature: purple-star: number -> image
  ```

- Remember: a **purpose statement** in Racket is written in a comment, and it **describes** what the function expects and **describes** what the function returns (and if the function has side-effects, it also **describes** those side-effects). For example,

  ```
  ; purpose: expects a size in pixels, the distance between
  ;    points of a 5-pointed-star, and returns an image
  ;    of a solid purple star of that size
  ```

- Remember: it is a COURSE STYLE STANDARD that named constants are to be descriptive and written in all-uppercase -- for example,

  ```
  (define WIDTH 300)
  ```

- You should use **blank lines** to separate your answers for the different parts of the homework problems. If you would like to add comments noting which part each answer is for, that is fine, too!

- **Because the design recipe is so important**, you will receive **significant** credit for the signature, purpose, header, and examples/check-expects portions of your functions. Typically you'll get at least half-credit for a correct signature, purpose, header, and examples/check-expects, even if your function body is not correct (and, you'll typically **lose at least half-credit** if you omit these or do them poorly, even if your function body is correct).

Problem 1

Start up DrRacket, (if necessary) setting the language to How To Design Programs - **Beginning Student** level, and adding the HTDP/2e versions of the image and universe teachpacks by putting these lines at the beginning of your Definitions window:

```
(require 2htdp/image)
(require 2htdp/universe)
```

Put a blank line, and then type in:

- a comment-line containing your name,
- followed by a comment-line containing `CS 111 - HW 2`,
- followed by a comment-line giving the date you last modified this homework,
- followed by a comment-line with no other text in it --- for example:

```
; type in YOUR name
; CS 111 - HW 2
; last modified: 2016-02-01
;
```

Below this, after a blank line, now type the comment lines:

```
;
; Problem 1
;
```

Now, consider a function that is to produce the perimeter of a rectangle.

1 part a

How many and what type of expressions should such a function expect, to be able to return such a perimeter?

Decide on a good, descriptive name for this function, and write an appropriate **signature** comment for this function.

1 part b

Write an appropriate **purpose statement** comment for this function.

1 part c

Write an appropriate **function header** for this function (putting `...)` for its body for now).

1 part d

Write at least 2 specific tests/`check-expect` expressions for this function. Interestingly, you may actually place these either before or after your not-yet-completed function definition, whichever you prefer.

1 part e

Only NOW should you replace the `...` in the function's body with an appropriate expression to complete this function.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 2

Next, in your definitions window, after a blank line, type the comment lines:

```
;
; Problem 2
;
```

Consider a function that asks a given person how they are doing. Given a person's name, it returns `How are you doing, ...that person's name... ?` (Optionally, you may change the exact wording of the question, as long as it includes the person's name and ends with at least one question mark.)

2 part a

How many and what type of expressions should such a function expect, to be able to return this question?

Decide on a good, descriptive name for this function, and write an appropriate **signature** comment for this function.

2 part b

Write an appropriate **purpose statement** comment for this function.

2 part c

Write an appropriate **function header** for this function (putting `... )` for its body for now).

2 part d

Write at least two specific tests/`check-expect` expressions for this function, placed either before or after your not-yet-completed function definition, whichever you prefer.

2 part e

Only NOW should you replace the `...` in the function's body with an appropriate expression to complete this function.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 3

Next, in your definitions window, after a blank line, type the comment lines:

```
;
```

```
; Problem 3
;
```

You decide it would be useful to be able to take a title and an image, and create a new image that includes the given title above that image, such that the title is depicted in 30-pixel-high red letters that are actually part of the image.

3 part a

Decide on a good, descriptive name for this function, and write an appropriate **signature** comment for this function.

3 part b

Write an appropriate **purpose statement** comment for this function.

3 part c

Write an appropriate **function header** for this function (putting `...)` for its body for now).

3 part d

Write at least two specific tests/`check-expect` expressions for this function, placed either before or after your not-yet-completed function definition, whichever you prefer.

3 part e

Only NOW should you replace the `...` in the function's body with an appropriate expression to complete this function.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 4

Next, in your definitions window, after a blank line, type the comment lines:

```
;
;  Problem 4
;
```

Consider a function `total-feet` that expects a number of miles and a number of feet, and returns the total number of feet. (For example, the value of the expression `(total-feet 10 5)` should be `52805`, because 10 miles and 5 feet is 52805 feet overall.)

4 part a

This part comes BEFORE starting function `total-feet`!

FIRST: write a Racket definition for a **named constant** that will define the **name** `FEET-PER-MILE` to be the number of feet in a mile. (`FEET-PER-MILE` **is NOT a function**! It is **just** a **number**, the number of feet in a **single** mile.)

4 part b

NOW you are starting the design recipe for function `total-feet`.

NOW, write an appropriate **signature** comment for the function `total-feet`.

4 part c

Write an appropriate **purpose statement** comment for `total-feet`.

4 part d

Write an appropriate **function header** for `total-feet` (putting `... )` for its body for now).

4 part e

Write at least two specific tests/`check-expect` expressions for `total-feet`,

- at least one for a number of miles of `0`, and
- at least one for a number of miles greater than `1`,
- (each with a DIFFERENT number of feet),

...placed either before or after your not-yet-completed function definition, whichever you prefer.

4 part f

Only NOW should you replace the `...` in `total-feet`'s body with an appropriate expression to complete it. For full credit, make sure that you use the named constant `FEET-PER-MILE` appropriately **within** this expression.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 5

Next, in your definitions window, after a blank line, type the comment lines:

```
;
;  Problem 5
;
```

(This is adapted from Homework 1 of J. Marshall's "The Way of the Program" course at Sarah Lawrence College.)

The Tasty-Waking coffee roasters sells whole-bean coffee at $8.75 per pound plus the cost of shipping. Each order ships for $0.75 per pound plus $2.50 fixed cost for overhead.

(So, if someone buys 1 pound of their whole-bean coffee, the total price is $12.00; if someone buys 2 pounds of their whole-bean coffee, the total price is $21.50; if someone buys 5 pounds, the total is $50.00.)

5 part a

The description above includes three constant, **unchanging** values. Define an appropriate, descriptive **named constant** for each of these values.

5 part b

Now consider a function `order-total` whose purpose is to calculate the total price for an order. Given that you already know the values for the named constants given in Problem 5 part a, how many and what type of **additional** expressions should such a function expect, to be able to calculate the price of a coffee order?

Write an appropriate **signature** comment for `order-total`.

(Hint: don't include named constants in the signature! Only consider what the function needs BESIDES the named constants to complete the task.)

5 part c

Write an appropriate **purpose statement** comment for this function.

5 part d

Write an appropriate **function header** for this function (putting `...)` for its body for now).

(Hint: remember that named constants do not belong in a function header, either.)

5 part e

Write at least two specific examples/tests/`check-expect` expressions for this function, placed either before or after your not-yet-completed function definition, whichever you prefer.

5 part f

Finally, replace the `...` in this function's body with an appropriate expression to complete it. For full credit, make sure that **NOW** you use the named constants from 5 part a appropriately **within** this expression.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 6

Next, in your definitions window, after a blank line, type the comment lines:

```
;
; Problem 6
;
```

Decide on a function that will take one or more parameters and return an image as a result. It can be anything not yet written in class or on a previous homework problem (variations on functions done in class or a homework problem are fine).

6 part a

Decide on a good, descriptive name for your function, and write an appropriate **signature** comment for this function.

6 part b

Write an appropriate **purpose statement** comment for this function.

6 part c

Write an appropriate **function header** for this function (putting `...`) for its body for now).

6 part d

Write at least two specific tests/`check-expect` expressions for this function, placed either before or after your not-yet-completed function definition, whichever you prefer.

6 part e

Only NOW should you replace the `...` in the function's body with an appropriate expression to complete this function.

Run and test your function until you are satisfied that it passes its tests and works correctly.

Finally, include at least 2 example calls of your function (such that you will see their results) after your function definition.

Problem 7

Oh yes -- and Problem 7 is correctly answering the CS 111 Syllabus Confirmation and Reading Questions from the course Moodle site by 11:59 pm on **Sunday, September 4**.